GameShop

M153-Projekt

Till Strasser, Louis Widmer

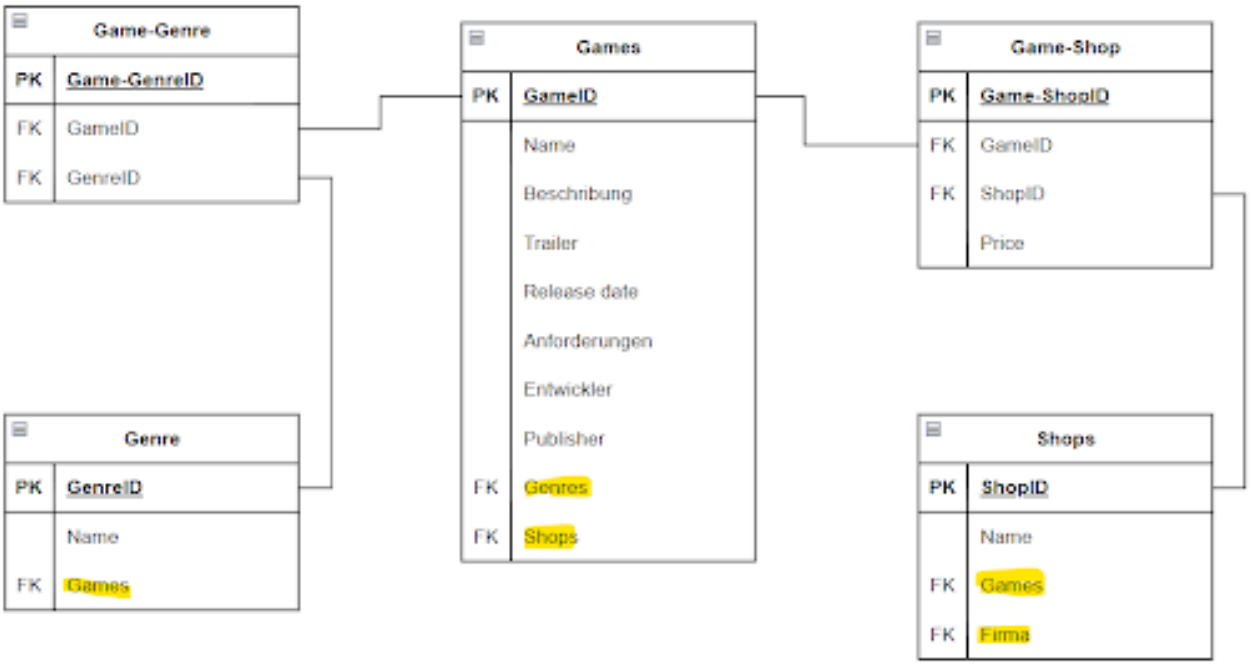
06.06.2023

Inhaltsverzeichnis

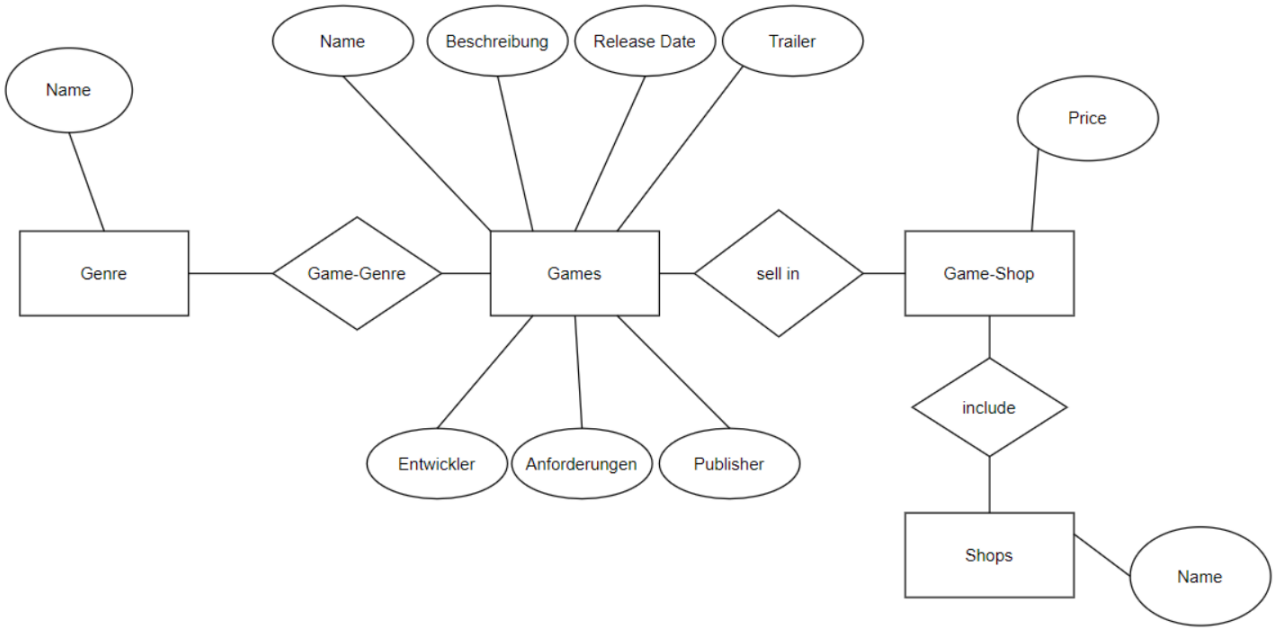
Kurzbeschreibung

Unsere Datenbank hat das Thema Gameshops, wir haben Spiele die verkauft werden, Shops wo man sie kaufen kann, Companys die Publisher und Developer haben und die Genres, die die Spiele haben können.

ERD-Diagramm



ERM-Diagramm



Beschreibung von den Functions

1.
```
--Function call: SELECT * FROM ComparePrices('Name of the Game')
create Function ComparePrices
(
    @gameName varchar(50)
)
returns @result table (
    ShopName varchar(50),
    Price int
)
as
begin
    insert into @result (ShopName, Price)
    select Shops.Name, Game_Shop.Price
    from Games
    JOIN Game_Shop on Games.GameID = Game_Shop.GameID
    JOIN Shops on Game_Shop.ShopID = Shops.ShopID
    where Games.Name = @gameName
    order by Game_Shop.Price asc;

    return;
end;

go
```

Man gibt ein Game an und dann werden die Preise von verschiedenen Shops verglichen.

2.
```
--Select * from SplitStrings('List of items like ('item 1, item 2')')
create Function SplitStrings
(
    @StringList varchar(50)
)
returns @result table (
    itemName varchar(50)
)
as
begin
    declare @itemName varchar(50);

    while len(@StringList) > 0
    begin
        if charindex(',', @StringList) > 0
        begin
            set @itemName = substring(@StringList, 1, charindex(',', @StringList)-1);
            set @StringList = substring(@StringList, charindex(',', @StringList)+1, len(@StringList));
        end
        else
        begin
            set @itemName = @StringList;
            set @StringList = '';
        end
        insert into @result (ItemName) values (@itemName);
    end
        return
end;

go
```

Diese Funktion holt sich Strings und splittet diese richtig, aus: “Item 1, Item 2” wird dann “Item 1”, “Item 2”

Beschreibung von Stored Procedures

1. Diese Procedure löscht alles, was aus Genres und Shops keine Verbindungen hat.

```
create procedure DeleteUnusedData
as
begin
        delete from Genres
        where GenreID not in (
                select GenreID from Game_Genre
        )

        delete from Shops
        where ShopID not in (
                select ShopID from Game_Shop
        )

end;
```

2. Diese Procedure fügt ein neues Game mit allen wichtigen Verbindungen hinzu

```
create procedure InsertNewGame
(
    @gameName varchar(50),
    @description varchar(500),
    @trailer varchar(100),
    @releaseDate date,
    @requirements varchar(500),
    @developer varchar(50),
    @publisher varchar(50),
    @genresList varchar(max),
    @shopsList varchar(max)
)
as
begin

        --insert Dev if its not in Company
        if @developer not in (select name from Company)
        begin
                insert into Company (Name)
                values (@developer)
        end
```

Selbsterklärender Code:

```
    --Inserting game informations
    insert into Games (Name, Description, Trailer, ReleaseDate, Requirements, DeveloperID, PublisherID)
    values (@gameName, @description, @trailer, @releaseDate, @requirements, (select top 1 CompanyID from Company where Name = @developer), (select top 1 CompanyID from Company where Name = @publisher));

    --Get the GameID
    declare @gameID int = @@IDENTITY;

    --Parse the list of genres into a table
    declare @genres table (
        GenreName varchar(50)
    );

        insert into @genres Select * from SplitStrings(@genresList)

        --insert genres if they do not exists
    insert into Genres (Name)
    select distinct GenreName from @genres
    where GenreName NOT IN (select Name from Genres);

    --conecting Game to Genres
    insert into Game_Genre (GameID, GenreID)
    select @gameID, GenreID
    from Genres
    where Name IN (SELECT GenreName from @genres);

    --Parse the list of shops
    declare @shops table (
        ShopName varchar(50)
    );

        insert into @shops Select * from SplitStrings(@shopsList)

        --insert Shops if they do not exists
    insert into Shops (Name)
    select distinct ShopName from @shops
    where ShopName NOT IN (select Name from Shops);

    --conecting Game to Shops
    insert into Game_Shop (Price, GameID, ShopID)
    select 0, @gameID, ShopID
    from Shops
    where Name IN (select ShopName from @shops);
end;
```

Diese Befehle gehören noch zum Insert Game Procedure, ist aber relativ selbsterklärend.

Tests

1. Hier wird getestet, ob das Hinzufügen von Genres funktioniert.

```
insert into genres (Name)
values ('Testgenre')
```

2. Hier wird getestet, ob das Hinzufügen von Shops funktioniert.

```
insert into Shops (Name)
values ('Testshop')
```

3. Hier wird getestet, ob man mit dem Testgenre, das man im 1. Test erstellt hat, zum Suchen nutzen kann.

```
select * from Genres
where Name = 'Testgenre'
```

4. Hier wird geschaut, ob mit dem vorher erstellten “Testshop” filtern kann.

```
select * from Shops
where Name = 'Testshop'
```

5. Hier wird geschaut, ob die Stored Procedure funktioniert.

```
exec DeleteUnusedData
```

6. Nachdem man diese Stored Procedure ausgeführt hat, sollten hierzu keine Ergebnisse mehr kommen.

```
select * from Genres
where Name = 'Testgenre'

select * from Shops
where Name = 'Testshop'
```

Der Database-Code

```
create table Company(
        CompanyID int identity(1, 1),
        Name varchar(50),
        primary key (CompanyID),
)

create table Games (
        GameID int identity(1,1),
        Name varchar(50),
        Description varchar(5000),
        Trailer varchar(100),
        ReleaseDate Date,
        Requirements varchar(5000),
        DeveloperID int,
        PublisherID int,
        primary key (GameID),
        constraint FK_Games_Developer foreign key (DeveloperID) references Company(CompanyID),
        constraint FK_Games_Publisher foreign key (PublisherID) references Company(CompanyID)
);

create table Game_Genre (
        GameID int,
        GenreID int,
        constraint PK_GameGenre primary key (GameID, GenreID),
        constraint FK_GameGenre_Game foreign key (GameID) references Games(GameID),
        constraint FK_GameGenre_Genre foreign key (GenreID) references Genres(GenreID)
);

create table Game_Shop(
        Price int,
        GameID int,
        ShopID int,
        constraint PK_GameShop primary key (GameID, ShopID),
        constraint FK_GameShop_Game foreign key (GameID) references Games(GameID),
        constraint FK_GameShop_Shop foreign key (ShopID) references Shops(ShopID)
);
```

```
use master

go

drop database if exists GamesShop
create database GamesShop

go

use GamesShop

go

create table Genres(
        GenreID int identity(1,1),
        Name varchar(50),
        primary key (GenreID)
);

create table Shops(
        ShopID int identity(1,1),
        Name varchar(50),
        primary key (ShopID)
);

create table Company(
        CompanyID int identity(1, 1),
        Name varchar(50),
        primary key (CompanyID),
)
```